

Exploiting Spatio-Temporal Structure with Recurrent Winner-Take-All Networks

Eder Santana¹, Matthew Emigh¹, Pablo Zerges², Jose C Principe¹, *Fellow, IEEE*

¹Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611 USA

² Facultad de Ingenieria y Ciencias Aplicadas, Universidad de los Andes, Chile

We propose a convolutional recurrent neural network, with Winner-Take-All dropout for high dimensional unsupervised feature learning in multi-dimensional time series. We apply the proposed method for object recognition with temporal context in videos and obtain better results than comparable methods in the literature, including the Deep Predictive Coding Networks previously proposed by Chalasani and Principe. Our contributions can be summarized as a scalable reinterpretation of the Deep Predictive Coding Networks trained end-to-end with backpropagation through time, an extension of the previously proposed Winner-Take-All Autoencoders to sequences in time, and a new technique for initializing and regularizing convolutional-recurrent neural networks.

Index Terms—deep learning, unsupervised learning, convolutional recurrent neural networks, winner-take-all, object recognition

I. INTRODUCTION

An elusive problem for both the cognitive and machine learning communities is the precise algorithm by which the human sensory system interprets the continuous stream of sensory inputs as stable perceptions of recognized objects and actions. In engineering, object and action recognition are tackled with supervised learning; on the other hand we have no evidence that the brain uses hard-wired or genetically evolved "supervisors" for training recognition networks. We argue that even if our nervous system applies supervised learning, the object classes should emerge in a self organizing way from experience and can be used as supervising labels.

It has been suggested that temporal information could be exploited as a possible source of supervision [1]. In this respect, suppose an observer is moving around and looking at an object in the middle of a room. Given that the input visual stream is continuous and that the object does not move, all the observed images of the object must belong to related perceptions and be represented in similar ways. This smooth, temporally coherent representation is biologically plausible and has been shown to self-organize V1-like Gabor filters when applied to learn image transition representations in video [2].

A machine learning application of this paradigm is to use temporal coherence as a proxy for learning sensory representations without strong supervision or explicit labels [3][4][5]. One approach is a Bayesian formulation in which we assume the learning system builds an internal model of the world $\tilde{p}(x_t; \theta)$ for explaining input streams x_t using a system parameterized by θ . Predictive Coding proposes to adapt this model to reduce discrepancies between predictions $\tilde{p}(x_t|x_{t-1}; \theta)$ and observations $p(x_t)$. Chalasani and Principe [6] developed a hierarchical, distributed, generative architecture called Deep Predictive Coding Networks (DPCN) that learns with free energy to build spatio-temporal shift-invariant representations of input video streams. They showed that DPCNs learned

features which can be used for classification, with competitive results albeit of being unsupervised. One of the difficulties of this approach is the required inference even in the testing state, which makes it rather slow.

This paper is inspired in DPCNs but substitutes the top down inference step in DPCN by a recurrent convolutional encoder-decoder that predicts the next frame of the video, can be trained with backpropagation, and does simple recall in test phase. The paper consists of three main contributions. First, we evolved the self-organizing object recognition in video work of Chalasani and Principe [6][7] by developing a scalable counterpart to the DPCN architecture and algorithms. Second, we build our contributions on top of recent findings in convolutional winner-take-all autoencoders [8] and convolutional-recurrent neural networks [9][10]. Thus, we extend the results of winner-take-all autoencoders to the time domain. We show results in video predictions, object recognition and action recognition. Third, Luong et. al. [11] showed that RNNs benefit from unsupervised pre-training and multi-task learning. We show that our method can be used as a pre-training technique for initializing convolutional RNNs.

To present the guiding principles for the proposed architecture, in the next section we overview DPCNs and point out its desirable features we would like to preserve and the undesirable features we would like to substitute.

II. DEEP PREDICTIVE CODING NETWORKS

Assume a multi-dimensional time series (e.g., a video) \mathbf{x}_t . Chalasani and Principe [6][7] proposed a generative, dynamical, hierarchical model with sparse coefficients \mathbf{s}_t :

$$\mathbf{s}_t = \mathbf{A}\mathbf{s}_{t-1} + \mathbf{v}_t \quad (1)$$

$$\mathbf{x}_t = \mathbf{C}\mathbf{s}_t + \mathbf{w}_t, \quad (2)$$

The sparsity of \mathbf{s}_t is controlled by a nonlinear higher order statistical component $\mathbf{B}\mathbf{u}_t$, where \mathbf{B} are the component weights and \mathbf{u}_t is itself L_1 -constrained to be sparse. This model can be stacked by generatively explaining, at layer l , the \mathbf{u}_t from the layer below it: $\mathbf{u}_t^l = \mathbf{C}\mathbf{s}_t^{l+1} + \mathbf{w}_t^{l+1}$. In practice,

This paper was partially funded by University of Florida Graduate Scholarship and ONR N00014-14-1-0542. Corresponding author: E. Santana (DM: <https://twitter.com/edersantana>).

this is accomplished by greedy layer-wise training. DPCNs are trained with Expectation-Maximization (EM) using the following energy function:

$$\mathcal{E}_t = \|\mathbf{x}_t - \mathbf{C}\mathbf{s}_t\|_2 + \lambda \|\mathbf{s}_t - \mathbf{A}\mathbf{s}_{t-1}\|_1 + \beta \|\mathbf{u}_t\|_1 + \sum_k^K |\mathbf{z}_t(k) \cdot \mathbf{s}_t(k)|, \quad (3)$$

$$\mathbf{z}_t = \gamma_0 \frac{1 + \exp(-\mathbf{B}\mathbf{u}_t)}{2}, \quad (4)$$

where the exponential function is applied to each element of the vector $-\mathbf{B}\mathbf{u}_t$, and k represents the vector indices.

Thus, given parameters \mathbf{C} , \mathbf{A} and \mathbf{B} , the DPCN algorithm searches for the \mathbf{s}_t that best fits an input \mathbf{x}_t . The term $\|\mathbf{s}_t - \mathbf{A}\mathbf{s}_{t-1}\|_1$ is a constraint that forces the solution to be as close as possible to a linear update of the solution for the previous input frame \mathbf{x}_{t-1} . This is an L_1 -slowness constraint to force temporal smoothness in the representation. The constraint $\sum_k^K |\mathbf{z}_t(k) \cdot \mathbf{s}_t(k)|$ forces the solution to be sparse, with sparsity level controlled by the higher level component \mathbf{u}_t . By doing so it also forces the network to learn time and space invariant features in the variable $\mathbf{z}_t(k)$.

DPCNs can be extended to handle large images by substituting the projection matrices \mathbf{A} , \mathbf{B} with convolutions and the latent codes \mathbf{s}_t , \mathbf{u}_t with feature maps, similarly to convolutional sparse coding [12]. Chalasani and Principe [7] used two layer convolutional DPCNs to extract features \mathbf{u}_t that were then used to train SVMs for classification surpassing several other sparse auto-encoding [13] and deconvolutional [14] techniques in accuracy on Caltech 101, Honda/UCSD faces [15], Celebrity Faces, and Coil-100 datasets.

Unfortunately, a few drawbacks exist that prevent DPCNs from scaling up and limit their application to other data domains. The DPCN uses expectation maximization to search for the codes \mathbf{s}_t , \mathbf{u}_t which best maximize the likelihood of input \mathbf{x}_t . This search is one of the reasons for DPCN's good results, but it is also a drawback. Even during test time it is necessary to execute computationally expensive expectation steps in order to estimate multidimensional coefficients \mathbf{s}_t , \mathbf{u}_t for each layer. Here instead we propose a parametrization using deep convolutional auto-encoders to compute the coefficients in a single operation for each layer.

Although successful as a model of the early stages of the visual system, imposing the multiplicative constraint $\mathbf{z}_t(k)$ limits the types of features learned in the higher layers of the architecture. In fact, previous experiments with DPCNs failed to learn better features with more than two layers, which is not usually the case for other deep learning architectures either supervised [16] or unsupervised [17].

Here instead, we will focus on learning deep representations from data using stacked rectified linear unity (ReLU) layers. This allows for simpler model building blocks which are still powerful enough for vision [16] and can be possibly extended to other datasets, such as audio [18].

DPCNs assume that given an input \mathbf{x}_t , the current latent code \mathbf{s}_t can be estimated given only the previous state \mathbf{s}_{t-1} and \mathbf{u}_t . Meanwhile, no backpropagation through time is carried

out to update the parameters \mathbf{A} , \mathbf{B} , \mathbf{C} . Although this might be interesting for learning in real-time without storing previous states, it limits the model from learning long-term dependencies. Also, without end-to-end connection the model cannot be fully fine-tuned with supervised learning, which has been shown to improve results on auto-encoders [19].

In the next section, the model we propose explicitly decouples spatial and temporal representations by learning state transitions in the latent space with RNNs. We keep the smoothness in time by forcing the RNN transitions to stay in a fixed radius open ball around the previous states.

Given DPCN's strengths and weaknesses, we attempt to develop an alternative architecture that is scalable, flexible and differentiable through time, while keeping as much as possible DPCN's abilities to learn discriminative statistics from spatio-temporal context.

III. RECURRENT WINNER-TAKE-ALL NETWORK

In this section we propose an end-to-end differentiable convolutional-recurrent neural network with Winner-Take-All dropout for feature extraction from video. In place of the linear state prediction matrix \mathbf{A} , we use Convolutional Recurrent Neural Networks (ConvRNNs) to predict future states. RNNs are particularly appropriate for this framework as they have a long history [20] of successfully modeling dynamical systems. Furthermore, in place of using computationally expensive EM algorithms to compute the sparse states and causes, we use convolutional autoencoders with Winner-Take-All [8] regularization to encode the states in feedforward manner.

Convolutional-recurrent neural networks [21][22] are RNNs where all the input-to-state and state-to-state transformations are implemented with convolutions. A vanilla-RNN state update can be represented in an equation as

$$\mathbf{h}_t = f(\mathbf{W}\mathbf{h}_{t-1} + \mathbf{V}\mathbf{x}_t), \quad (5)$$

where the dynamic state vector \mathbf{h}_t represents the history of the time series up to time t , and f is a nonlinearity such as the hyperbolic tangent or ReLU. In a convolutional RNN, the dynamic state is a 3-way tensor $\mathbf{h}_{t,f,r,c}$ with f channels, r rows and c columns. The hidden-to-hidden transition operation is a convolutional kernel \mathbf{W}_{o,f,r_w,c_w} with $r_w < r$ and $c_w < c$, and $o = f$ output channels. Similarly, we have a convolutional kernel for the input-to-hidden transition $\mathbf{V}_{f,f_x,r_w,c_w}$, where f_x is the number of channels in the input $\mathbf{x}_{t,f_x,r,c}$.

$$\mathbf{h}_{t,f,r,c} = f(\mathbf{W}_{f,f,r_w,c_w} \star \mathbf{h}_{t-1,f,r,c} + \mathbf{V}_{f,f_x,r_w,c_w} \star \mathbf{x}_{t,f_x,r,c}), \quad (6)$$

where \star denotes the multi-channel convolution operator used in deep learning:

$$(W \star h)_{fij} = \sum_{a,b,c} \mathbf{W}_{f,a,b,c} \mathbf{h}_{a,i-b,j-c}. \quad (7)$$

A schematic representation of convolutional RNN is shown in Figure 1.

RNNs can be trained in several ways for sequence prediction [20][23]; here we focus on training our ConvRNN to predict the next frame of the sequence. Overfitting in conventional RNNs is avoided using bottleneck layers, where the RNN

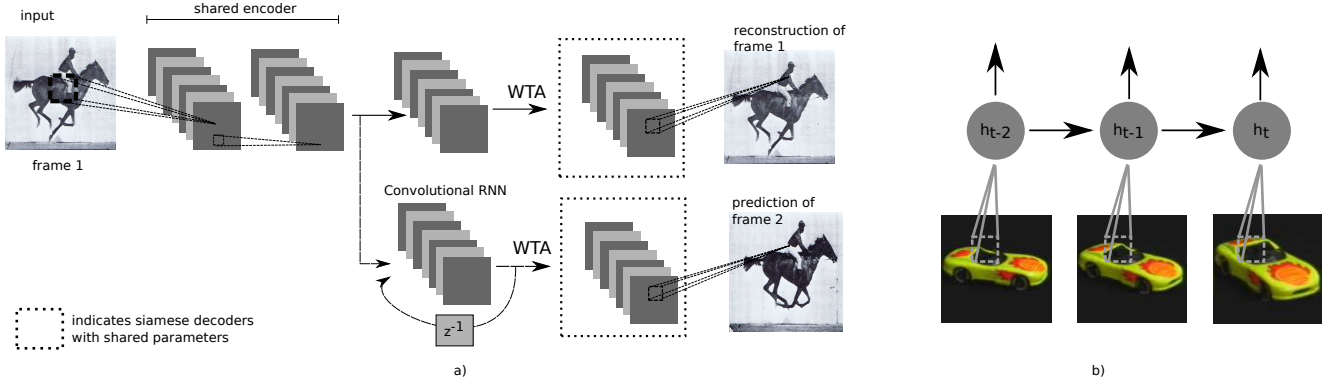


Fig. 1. Left: Schematic diagram of the proposed two-stream convolutional recurrent neural networks architecture with Winner-Take-All (RWTA) regularization. Upper stream is the static encoder-decoder. Lower stream denotes the temporal, dynamic encoder. Right: Representation of a convolutional recurrent neural network unfolded in time for a fixed spatial location. The same recurrent neural network is applied throughout the entire image.

hidden state dimensions are smaller than the input. On the other hand, as discussed in the DPCN section, for image analysis we want to expand the latent space dimensionality and avoid overfitting with sparseness constraints. The advantages of high dimensional representations are formalized by Cover's theorem.

In autoencoders, sparsity can be imposed as a constraint on the objective function [24]. Unfortunately, sparseness as measured by L_1 or L_0 norms is hard to optimize, requiring elaborate techniques such as proximal gradients (ex. FISTA [25]) or learned approaches (ex. LISTA [26]). Since we want sparseness in the network outputs and not network weights, those techniques would also require optimization during test time, as done in the original formulation of DPCNs. Recent research has shown that simple regularization techniques such as Dropout [27] combined with ReLU activations are enough to learn sparse representations without extra penalties in the cost functions. This represents a paradigm shift from cost function to architectural regularization that provides faster training and testing.

Makhzani and Frey proposed Winner-Take-All (WTA) Autoencoders [8] which use aggressive Dropout, where all the elements but the strongest of a convolutional map are zeroed out. This forces sparseness in the latent codes and the convolutional decoder to learn robust features. Here we extend convolutional Winner-Take-All autoencoders through time using convolutional RNNs. WTA for a map $x_{f,r,c}$ in the output of convolutional layer can be expressed as in (8). The indices f, r, c represent respectively the number of rows, the number of columns, and the number of channels in the map.

$$WTA(x_{f,r,c}) = \begin{cases} x_{f,r,c}, & \text{if } x_{f,r,c} = \max_{r,c}(x_{f,r,c}) \\ 0, & \text{otherwise} \end{cases} \quad (8)$$

Thus, $WTA(x_{f,r,c})$ has only one non-zero value for each channel f . To backpropagate through (8) we use $\nabla WTA(x_{f,r,c}) = WTA(\nabla x_{f,r,c})$. In the present paper, we apply (8) to the output of the convolutional maps of the ConvRNNs after they have been calculated. In other words, the full convolutional map hidden state is used inside the dynamics

of the ConvRNN, WTA is applied only before they are fed as input to the convolutional decoder. We leave investigations of how WTA would affect the inner dynamics of ConvRNNs for future work.

We propose to learn smoothness in time with architectural constraints using a two-stream encoder as shown in Figure 1. This architecture was inspired by the dorsal and ventral streams hypothesis in the human visual cortex [28]. Roughly speaking, the dorsal stream models "vision for action" and movements and the ventral stream represents "vision for perception". In our proposed architecture one stream is a stateless convolutional encoder-decoder and the other stream has a convolutional RNN encoder, thus a dynamic state. Using Siamese decoders for both streams, we force the stateless encoder and the convolutional RNN to project into the same space—one which can be reconstructed by the shared weights decoder. It is important to stress that from the point of view of spatio-temporal feature extraction with the ConvRNN, the stateless stream works as for regularization. As any other sort of regularization its usefulness can only be totally stated in practice and the practitioner might optionally not use it. Nevertheless, we opted for using the full architecture in all the experiments of this paper. In Appendix A, we show how this proposed architecture enforces spatio-temporal smoothness in the embedded space.

Given an input video stream \mathbf{x}_t , denoting the stateless encoder by E , the decoder D , and the convolutional RNN by R , the cost function for training our architecture is the sum of reconstruction and prediction errors:

$$L_t = \mathbb{E} [(\mathbf{x}_{t-1} - D(E(\mathbf{x}_{t-1})))^2 + (\mathbf{x}_t - D(R(\mathbf{x}_{t-1})))^2], \quad (9)$$

where \mathbb{E} denotes the expectation operator. Notice that as depicted in Figure 1, E and R have shared parameters. During training, we observe a few input frames $t = [1, 2, \dots, T]$ and adapt all the parameters using backpropagation through time (BPTT) [29]. Notice that due to BPTT both streams of our architecture are adapted while considering temporal context. Thus, the stateless encoder E will learn richer features than it would if trained on individual frames.

As great power brings great responsibility, the main draw-

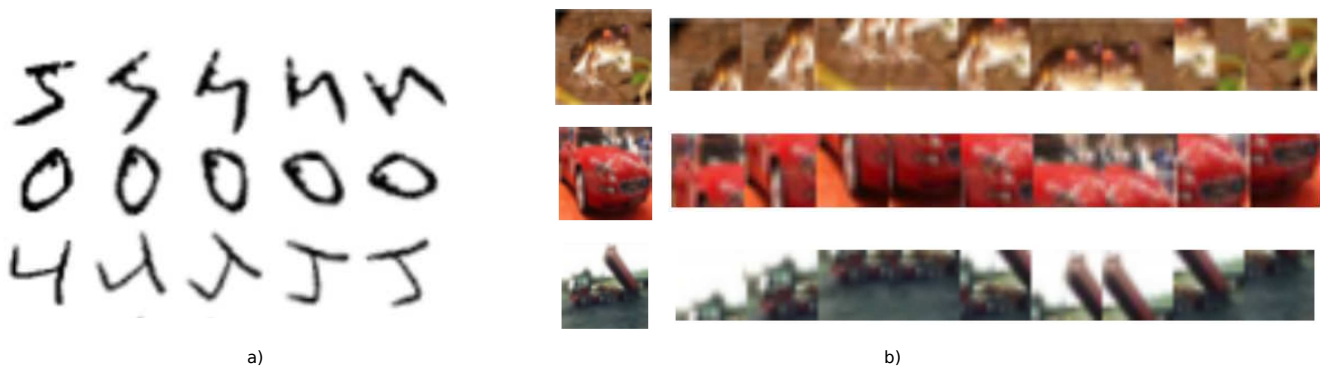


Fig. 2. Left: Sample videos of the rotated MNIST dataset. Right: Sample original images and videos generated by scanning the images with 16x16 pixels windows. The scanning runs vertical first, starting in the upper left corner.

back of our proposed architecture is the memory required by BPTT and convolutions. The gradients of convolutions require storing the multi-way tensor output of the convolutions, and BPTT requires storing all the outputs for all time steps. The combination of both methods in a single architecture requires powerful hardware. We limited the length of our input time series between 5 and 10 frames, which is also the length used by the methods with which we compare in the Experiments section. In the next section, we compare our proposed architecture with similar methods proposed in the literature beyond the already discussed DPCN.

IV. RELATED WORK

This research is related to DPCNs and a larger family of deep unsupervised neural networks [24][30][17]. The aforementioned Winner-Take-All Autoencoders (WTA-AE) [8] consist of a deep convolutional encoder and a single layer convolutional decoder, which inspired our choice. WTA-AE drops out all the elements of a convolutional channel map but the largest, forcing the whole system to learn robust, sparse features for reconstruction. With the proposed convolutional RNN our method can be seen as a natural extension of WTA-AE.

Unsupervised learning with temporal context was also previously explored by Goroshin et. al. [3] Wang and Gupta [31]. Their approach was based on metric learning of related frames in video, but their approaches were not capable of learning long term dependencies since they assumed only a simple zero-mean Gaussian innovation between frames. Also, neither of these approaches can be fine-tuned by BPTT to learn end-to-end classifiers in time.

Convolutional RNNs were proposed simultaneously by several authors [22][21] as an extension of Network-in-Networks [32] architectures where each convolutional layer of a CNN are themselves deep networks. Liang et. al [22] proposed to make each convolutional layer a fixed input RNN. They used that architecture for object recognition in static scenes without exploring temporal context. Xingjian et. al. [21] and Patraucean et. al. [33] on the other hand used temporal context in videos for weather and video forecasts. Their architectures are similar to predictive networks, but they did not address the problem of regularizing the latent space features, nor how

to train deep architectures—their models consist only of a single convolutional RNN module for predicting future frames. Furthermore, they do not investigate how to extract interesting features without context, which is the problem addressed by our stateless encoder-decoder stream trained in parallel with the dynamic stream.

In parallel to this paper, another follow up on the DPCN approach was published by Bill Lotte et. al [34]. Differently from this paper, their method, called PredNet, focused on frame prediction for video and not sparse feature extraction. Nevertheless, both approaches are complementary and could be combined in future work.

V. EXPERIMENTS

To illustrate the capabilities of our proposed architecture we applied it first to two artificial datasets generated by modifying the MNIST and Cifar10 datasets. We used MNIST and Cifar10 as development datasets to understand how hyperparameter choices affect our method; that is, to understand how many filters per layer are necessary, how much temporal context contributes to learning unsupervised features, and how long to take in the unsupervised phase. The full list of hyperparameters is shown on Table I. Note that we fixed the number of channels per convolutional layer be equal in layers to limit the number of hyperparameters. An exception is the number of channels in the decoder (the very last layer) since it has to match the number of channels in the input (i.e. 1 for black and white images and 3 for color images).

Furthermore, for the modified MNIST dataset, we show our architecture learns, using temporal information, more discriminative features. For the modified Cifar10 dataset, we show the advantage of pre-training convolutional RNNs with our method.

Afterwards, we applied our best performing architectures to the Coil100 and Honda/UCSD Faces Dataset for a direct comparison with DPCN and other unsupervised learning techniques.

A. Rotated MNIST Dataset

We extended the MNIST dataset by generating videos by rotating each image counter-clockwise. Sample videos are

TABLE I
HYPERPARAMETER CHOICES PER EXPERIMENT

	rotated MNIST	scanned Cifar10	COIL100	Honda Faces
Channels per layer	64	256	128	256
Filter size (encoder)	3x3	5x5	5x5	5x5
Filter size (decoder)	11x11	7x7	7x7	7x7
All models were trained using ADAM optimization rule with learning rate 0.001				
All models were 4 layers deep				
All models had 2 convolutional layers before the ConvRNN layer.				
WTA was applied only right before the last layer.				

shown in Fig. 2. We trained our two-stream convolutional RNN on videos generated with MNIST training dataset. The task was to learn to reconstruct and predict frames as described in 9. We trained the networks with batches of size 100 for 3 epochs (a total of 1800 updates) using the Adam learning rule [35]. We trained a linear Support Vector Machine (SVM) on the features computed by the convolutional RNN R . We collapsed the temporal features into one using addition: $z = \sum_t R(x_t)$, where the z 's are the input to train the SVM. All the encoder convolutional kernels had $f = 64$ channels of size $c = r = 3$. The classification error probability on videos generated with the MNIST test set was 0.94%. An equivalent WTA-AE obtained only 1.02% accuracy.

We argue that the possible reasons for the better performance of the proposed method are due to data augmentation and the capacity of the proposed method to use that augmentation to compose a single, less ambiguous interpretation of the data. In Figure 2 we show the 64 filters of 11x11 pixels learned by the decoder D .

B. Scanned Cifar10 dataset

The Cifar10 dataset consists of 50k RGB images of 32x32 pixels for training and an additional 10k images for testing. There is a total of 10 different classes. We converted this dataset to videos by scanning it with 16x16 windows that move 8 pixels at a time as shown in Fig. 2. The 16x16 windows were pre-processed with ZCA. In most of the videos no single 16x16 window completely captures the object to be classified. This forces a classifier to use "temporal" context to perform well.

We trained the proposed method on this dataset for 10 epochs. Each convolutional map of the encoders E and R had 256 filters of size 5x5 pixels. The decoder had filters of size 7x7. We then fine-tuned the convolutional RNN for classification using supervised learning and obtained a classification rate of 75.6%, while a similar convolutional RNN trained from scratch obtained only 74.1%. Both networks were equally initialized using the Glorot uniform method [36].

We did not have success using a single linear SVM on sum-collapsed features for this dataset. Nevertheless, this experiment suggests that even when the proposed pre-training technique in itself is not enough for learning relevant features, it can still be used as an initialization technique for complex recurrent systems.

Using what we learned with these two preliminary examples on the modified MNIST and Cifar10 datasets we decided to use the following general guidelines for the following experiments: 1) Color videos are preprocessed by ZCA. 2) Convolutional filters use either 128 or 256 channels of size 3x3 or 5x5 in the encoder and 7x7 in the decoder. 3) Training takes ≈ 1500 updates. For our dataset this was about 10 epochs long, with batch sizes of 16 or 32, depending on GPU memory. 4) Linear SVM classifiers were trained on encoded version of the last frame of a sequence $R(x_T)$. In our experiments $T = 5$. For longer videos in test time, we classified each frame using by moving the $T = 5$ window one frame at a time and took the most voted class as the final guess.

C. Coil-100 Dataset

The COIL-100 dataset [37] consists of 100 videos of different objects. Each video is 72 frames long and were generated by placing the object on a turn table and taking a picture every 5° . The pictures are 128x128 pixels RGB. For our experiments, we rescaled the images to 32x32 pixels and used ZCA pre-processing.

The classification protocol proposed in the COIL-100 [37] uses 4 frames per video as labeled samples, the frames corresponding to angles $0^\circ, 90^\circ, 180^\circ$ and 270° . Chalasani and Principe[7] and Mobahi et. al. [38] used the entire dataset for unsupervised pre-training. For this reason, we believe the results in this experiment should be understood with this in mind. Note that the compared methods enforce smoothness in the representation of adjacent frames, and since the test frames are observed in context for feature extraction, information is carried from labeled to unlabeled samples. In other words, this experiment is better described as semi-supervised metric learning than unsupervised learning. Here, we followed that same protocol, using 14 frames per video. Results are reported in Table II. We used encoders with 128 filters of 5x5 pixels and a decoder with 7x7 pixels. The decoder filters are shown in Fig. 3

D. Honda/UCSD Dataset

The Honda/UCSD dataset consists of 59 videos of 20 different people moving their heads in various ways. The training set consists of 20 videos (one for each person), $\sim 300-1000$ frames each. The test set consists of 39 videos (1-4 per person), $\sim 300-500$ frames each. For each frame of all

TABLE III
RECOGNITION RATE (IN PERCENTAGE %) FOR FACE RECOGNITION IN HONDA/UCSD DATASET

Sequences Lengths	MDA [40]	SANP [41]	CDN [7]	Proposed Method
50 Frames	74.36	84.62	92.31	100
100 Frames	94.87	92.31	100	100
Full Video	97.44	100	100	100

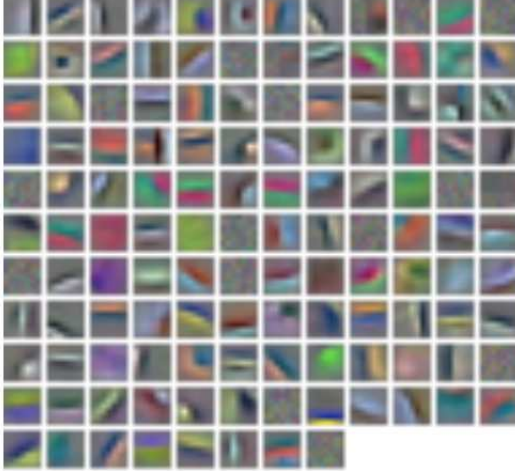


Fig. 3. 128 decoder weights of 7x7 pixels learned on Coil-100 videos.

TABLE II
RECOGNITION RATE (IN PERCENTAGE %) FOR OBJECT RECOGNITION IN COIL-100 DATASET

Method	Accuracy
DPCN no context [7]	79.45
Stacked ISA + temporal [39]	87
ConvNets + Temporal [38]	92.25
DPCN + temporal + top down [7]	98.34
Proposed method	99.4

videos, we detected and cropped the faces using Viola-Jones face detection. Each face was then converted to grayscale, resized to 20x20 pixels, and histogram equalized.

During training, the entire training set was fed into the network, 9 frames at a time, with a batch size of 32. After training was complete, the training set was again fed into the network. For each input frame in the sequence, the feature maps from the convolutional RNN were extracted, and then (5,5) max-pooled with a stride of (3,3). In accordance with the test procedure of Chalasani and Principe [7], a linear SVM was trained using these features and labels indicating the identity of the face. Finally, each video of the test set was fed into the network, one frame at a time, and features were extracted from the RNN in the same way as described above. Each frame was then classified using the linear SVM. Each sequence was assigned a class based on the maximally pooled predicted label across each frame in the sequence. Table III summarizes the results for 50 frames, 100 frames, and the full video, comparing with 3 other methods, including the original

convolutional implementation of DPCN [7]. The results for the 3 other methods were taken from [7]. The results for our method were perfect for all the tested cases.

VI. PRELIMINARY RESULTS AND FUTURE WORK

In all our experiments we investigate single scale feature extraction, i.e., we did not use pooling or strided convolutions. In experiments not discussed in this paper, we explored pooling and unpooling in the proposed architecture. However, this did not improve the results considerably. Nevertheless, Makhzani and Frey [8] showed that layer wise training at different scales improved their results. They first trained a convolutional WTA-AE on the raw data, downsampled the features with maxpooling and trained another layer on top the pooled features. Learning at a different scale could be done in two different ways with the proposed architecture. The first way would be by collapsing the temporal features into a single feature map (by addition or picking the last state), downsampling and learning a WTA-AE on top for the resulting features. The second approach would be to downsample every feature in time and train a second two-stream convolutional RNN. Which approach is best remains elusive, but we plan to investigate further in future work.

In our experiment sections, we showed how the proposed architecture compares favorably to other methods that leverage temporal context for object recognition. Future experiments should focus on problems such as action recognition, where the best performance cannot be achieved using single frame recognition. In preliminary experiments, we applied the same architecture used with the Coil-100 dataset to the UCF-101 action recognition dataset. A linear SVM applied to unsupervised features was used to obtain a recognition rate of 44% which is slightly better than the baseline of 43.90%. On the other hand, methods based on supervised learning with deep convolutional neural networks using hand-engineered spatial flow features can obtain a recognition rates $> 80\%$ [42][43]. Our follow up work should investigate how to learn unsupervised features with architectural constraints similar to those used for calculating spatial flow, e.g. local differences in the pixel space.

When investigating the benefits of unsupervised training for discriminative convolutional RNNs, we only showed results for unsupervised initialization. However, Luong et. al. [11] showed that multitask learning can improve the overall results of the supervised learning task. In other words, while training a conventional RNN for classification, they also added an unsupervised term to the objective function. They argued that such multitask learning regularized the RNN and avoided

overfitting. In convolutional architectures the number of latent features (outputs of convolutional layers) is much larger than the number of learned parameters, which already enforces some regularization. However, given that natural images are highly correlated in local neighborhoods, we believe that multitask learning will also benefit convolutional RNNs trained for supervised tasks.

VII. CONCLUSIONS

This paper proposes RWTa, a deep convolutional recurrent neural network with Winner-Take-All dropout for extracting features from time series. Our contributions were threefold: 1) a scalable, end-to-end differentiable reinterpretation of the sparse spatio-temporal feature extraction in Deep Predictive Coding Networks [7]; 2) an extension of Winner-Take-All Autoencoders [8] to time using dynamic neural networks; 3) a new technique for initializing and regularizing [11] convolutional recurrent neural networks [9][10]. We showed that our method outperforms DPCNs and other similar methods in contextual object recognition tasks. We also showed that this method can be used as an initialization technique for supervised convolutional RNNs, obtaining better results than Glorot initialization [36].

APPENDIX A

RWTa LEARNED INVARIANCES

Assume input stream \mathbf{x}_t , where t is a countable index, is fed into two modules, a static and a recurrent neural network respectively:

$$\mathbf{o}_t^E = \mathbf{e}(\mathbf{x}_t) \quad (10)$$

$$\mathbf{o}_t^R = \mathbf{r}(\mathbf{x}_t, \mathbf{o}_{t-1}), \quad (11)$$

where \mathbf{r} is a recurrent neural network and the state \mathbf{o}_{t-1} gives it context for prediction. These two outputs are fed into a siamese decoders that produces another two outputs

$$\mathbf{f}_t^E = \mathbf{d}(\mathbf{o}_t^E) \quad (12)$$

$$\mathbf{f}_t^R = \mathbf{d}(\mathbf{o}_t^R) \quad (13)$$

Training is done such that the following expression is minimized:

$$E = \sum_t (\mathbf{x}_{t-1} - \mathbf{f}_{t-1}^E)^2 + \sum_t (\mathbf{x}_t - \mathbf{f}_{t-1}^R)^2 \quad (14)$$

The main effect of the WTA algorithm [?] when applied to \mathbf{o}_t^E and \mathbf{o}_t^R , is to partition the input space of the corresponding functions into volumes that produce the same output. Hence, if there is a sequence of inputs in the interval $\{t, \dots, t+k\}$ that is contained inside one of these volumes, then:

$$\mathbf{o}_t^E = \mathbf{o}_{t+1}^E = \mathbf{o}_{t+2}^E = \dots = \mathbf{o}_{t+k}^E = \mathbf{e}(\mathbf{x}_t) = \mathbf{e}(\mathbf{x}_{t+1}) = \dots = \mathbf{e}(\mathbf{x}_{t+k}) \quad (15)$$

$$\mathbf{o}_t^R = \mathbf{o}_{t+1}^R = \mathbf{o}_{t+2}^R = \dots = \mathbf{o}_{t+k}^R = \mathbf{r}(\mathbf{x}_t) = \mathbf{r}(\mathbf{x}_{t+1}) = \dots = \mathbf{r}(\mathbf{x}_{t+k}) \quad (16)$$

which implies

$$\mathbf{f}^E = \mathbf{f}_t^E = \mathbf{f}_{t+1}^E = \dots = \mathbf{f}_{t+k}^E = \mathbf{d}(\mathbf{o}^E) \quad (17)$$

$$\mathbf{f}^R = \mathbf{f}_t^R = \mathbf{f}_{t+1}^R = \dots = \mathbf{f}_{t+k}^R = \mathbf{d}(\mathbf{o}^R) \quad (18)$$

Hence, isolating the corresponding section of the objective function, and using the previous equalities, the following expression needs to be minimized:

$$\begin{aligned} E_{t,t+k} &= (\mathbf{x}_t - \mathbf{f}_t^E)^2 + (\mathbf{x}_{t+1} - \mathbf{f}_t^R)^2 \\ &\quad + (\mathbf{x}_{t+1} - \mathbf{f}_{t+1}^E)^2 + (\mathbf{x}_{t+2} - \mathbf{f}_{t+1}^R)^2 \\ &\quad \dots \\ &\quad + (\mathbf{x}_{t+k} - \mathbf{f}_{t+k}^E)^2 + (\mathbf{x}_{t+k+1} - \mathbf{f}_{t+k}^R)^2 \end{aligned} \quad (19)$$

$$\begin{aligned} &= (\mathbf{x}_t - \mathbf{f}^E)^2 + (\mathbf{x}_{t+1} - \mathbf{f}^R)^2 \\ &\quad + (\mathbf{x}_{t+1} - \mathbf{f}^E)^2 + (\mathbf{x}_{t+2} - \mathbf{f}^R)^2 \\ &\quad \dots \end{aligned}$$

$$\begin{aligned} &\quad + (\mathbf{x}_{t+k} - \mathbf{f}^E)^2 + (\mathbf{x}_{t+k+1} - \mathbf{f}^R)^2 \end{aligned} \quad (20)$$

$$\begin{aligned} &= (\mathbf{x}_t - \mathbf{d}(\mathbf{o}^E))^2 + (\mathbf{x}_{t+1} - \mathbf{d}(\mathbf{o}^R))^2 \\ &\quad + (\mathbf{x}_{t+1} - \mathbf{d}(\mathbf{o}^E))^2 + (\mathbf{x}_{t+2} - \mathbf{d}(\mathbf{o}^R))^2 \\ &\quad \dots \end{aligned}$$

$$\begin{aligned} &\quad + (\mathbf{x}_{t+k} - \mathbf{d}(\mathbf{o}^E))^2 + (\mathbf{x}_{t+k+1} - \mathbf{d}(\mathbf{o}^R))^2 \end{aligned} \quad (21)$$

Thus we can do the following considerations. Minimization of the last expression shows that $\mathbf{d}(\mathbf{o}^E)$ and $\mathbf{d}(\mathbf{o}^R)$ will try to be close to all the $\mathbf{x}_t, \mathbf{x}_{t+1}, \dots, \mathbf{x}_{t+k}$ in a square sense. Hence it will produce $\mathbf{f}_*^E = \mathbf{d}(\mathbf{o}^E)$ and $\mathbf{f}_*^R = \mathbf{d}(\mathbf{o}^R)$. Moreover, neglecting the two extreme terms of the last expression shows that it is composed of pairs of sums such as $(\mathbf{x}_{t+1} - \mathbf{d}(\mathbf{o}^R))^2 + (\mathbf{x}_{t+1} - \mathbf{d}(\mathbf{o}^E))^2$. Thus the minimization process is also trying to make $\mathbf{d}(\mathbf{o}^R) \approx \mathbf{d}(\mathbf{o}^E)$ which implies $\mathbf{o}^R \approx \mathbf{o}^E$. This forces a balance between the stateless encoder and the recurrent neural network. Hence the system will look for solution that consider spatial and temporal invariances. Given that the capacity of the system to partition the space is limited, there is a limited number of \mathbf{o}^R and \mathbf{o}^E encodings that the system can produce. Thus the training procedure will focus on finding those volume partitions that can be used to explain the incoming stream in an efficient manner. The combined effect of all these components finally act like an invariance detector.

REFERENCES

- [1] R. Baker, M. Dexter, T. E. Hardwicke, A. Goldstone, and Z. Kourtzi, "Learning to predict: Exposure to temporal sequences facilitates prediction of future events," *Vision Research*, vol. 99, pp. 124 – 133, 2014.
- [2] J. Hurri and A. Hyvärinen, "Temporal coherence, natural image sequences, and the visual cortex," in *Advances in Neural Information Processing Systems*, 2002, pp. 141–148.
- [3] R. Goroshin, J. Bruna, J. Tompson, D. Eigen, and Y. LeCun, "Unsupervised learning of spatiotemporally coherent metrics," in *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.
- [4] X. Wang and A. Gupta, "Unsupervised learning of visual representations using videos," *CoRR*, vol. abs/1505.00687, 2015. [Online]. Available: <http://arxiv.org/abs/1505.00687>
- [5] P. Agrawal, J. Carreira, and J. Malik, "Learning to see by moving," *CoRR*, vol. abs/1505.01596, 2015. [Online]. Available: <http://arxiv.org/abs/1505.01596>
- [6] R. Chalasani and J. Principe, "Deep predictive coding networks," in *Workshop at International Conference on Learning Representations (ICLR)*, 2013.

- [7] R. Chalasani and J. C. Principe, "Context dependent encoding using convolutional dynamic networks," *Neural Networks and Learning Systems, IEEE Transactions on*, vol. 26, no. 9, pp. 1992–2004, 2015.
- [8] A. Makhzani and B. J. Frey, "Winner-take-all autoencoders," in *Advances in Neural Information Processing Systems*, 2015, pp. 2773–2781.
- [9] M. Liang and X. Hu, "Recurrent convolutional neural network for object recognition," in *CVPR*, June 2015.
- [10] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," *arXiv preprint arXiv:1506.04214*, 2015.
- [11] M. Luong, Q. V. Le, I. Sutskever, O. Vinyals, and L. Kaiser, "Multi-task sequence to sequence learning," *CoRR*, vol. abs/1511.06114, 2015. [Online]. Available: <http://arxiv.org/abs/1511.06114>
- [12] J. Yang, K. Yu, and T. Huang, "Supervised translation-invariant sparse coding," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 3517–3524.
- [13] K. Kavukcuoglu, P. Sermanet, Y.-L. Boureau, K. Gregor, M. Mathieu, and Y. L. Cun, "Learning convolutional feature hierarchies for visual recognition," in *Advances in neural information processing systems*, 2010, pp. 1090–1098.
- [14] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus, "Deconvolutional networks," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 2528–2535.
- [15] K. Lee, J. Ho, M. Yang, and D. Kriegman, "Video-based face recognition using probabilistic appearance manifolds," *IEEE Conf. On Computer Vision and Pattern Recognition*, vol. 1, pp. 313–320, 2003.
- [16] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.
- [17] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009, pp. 609–616.
- [18] M. D. Zeiler, M. Ranzato, R. Monga, M. Mao, K. Yang, Q. V. Le, P. Nguyen, A. Senior, V. Vanhoucke, J. Dean *et al.*, "On rectified linear units for speech processing," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 3517–3521.
- [19] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.
- [20] A. Graves, "Generating sequences with recurrent neural networks," *arXiv preprint arXiv:1308.0850*, 2013.
- [21] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-k. Wong, and W.-c. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," in *Advances in Neural Information Processing Systems*, 2015, pp. 802–810.
- [22] M. Liang and X. Hu, "Recurrent convolutional neural network for object recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3367–3375.
- [23] I. Sutskever, "Training recurrent neural networks," Ph.D. dissertation, University of Toronto, 2013.
- [24] B. A. Olshausen *et al.*, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images," *Nature*, vol. 381, no. 6583, pp. 607–609, 1996.
- [25] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM journal on imaging sciences*, vol. 2, no. 1, pp. 183–202, 2009.
- [26] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010, pp. 399–406.
- [27] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.
- [28] M. A. Goodale and A. D. Milner, "Separate visual pathways for perception and action," *Trends in neurosciences*, vol. 15, no. 1, pp. 20–25, 1992.
- [29] P. J. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.
- [30] M. Noroozi and P. Favaro, "Unsupervised learning of visual representations by solving jigsaw puzzles," *arXiv preprint arXiv:1603.09246*, 2016.
- [31] X. Wang and A. Gupta, "Unsupervised learning of visual representations using videos," in *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec 2015, pp. 2794–2802.
- [32] M. Lin, Q. Chen, and S. Yan, "Network in network," *CoRR*, vol. abs/1312.4400, 2013. [Online]. Available: <http://arxiv.org/abs/1312.4400>
- [33] V. Patraucean, A. Handa, and R. Cipolla, "Spatio-temporal video autoencoder with differentiable memory," *arXiv preprint arXiv:1511.06309*, 2015.
- [34] W. Lotter, G. Kreiman, and D. Cox, "Deep predictive coding networks for video prediction and unsupervised learning," *arXiv preprint arXiv:1605.08104*, 2016.
- [35] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
- [36] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *International conference on artificial intelligence and statistics*, 2010, pp. 249–256.
- [37] S. A. Nene, S. K. Nayar, H. Murase *et al.*, "Columbia object image library (coil-20)," technical report CUCS-005-96, Tech. Rep., 1996.
- [38] H. Mobahi, R. Collobert, and J. Weston, "Deep learning from temporal coherence in video," in *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009, pp. 737–744.
- [39] Q. V. Le, W. Y. Zou, S. Y. Yeung, and A. Y. Ng, "Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis," in *CVPR*. IEEE, 2011, pp. 3361–3368.
- [40] R. Wang and X. Chen, "Manifold discriminant analysis," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 429–436.
- [41] Y. Hu, A. S. Mian, and R. Owens, "Sparse approximated nearest points for image set classification," in *Computer vision and pattern recognition (CVPR), 2011 IEEE conference on*. IEEE, 2011, pp. 121–128.
- [42] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Advances in Neural Information Processing Systems*, 2014, pp. 568–576.
- [43] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," *arXiv preprint arXiv:1412.0767*, 2014.